Welcome to JavaScript Function series. In part 1, we discussed Function Declaration syntax in JavaScript and few traps associated with it. In this article we will discuss JavaScript **Function Expression**.

**Function Expression**

A function expression has similar syntax as function declaration except that function value is assigned to a variable name.

var log = function logMessage(message){

console.log(message);

}

log("This is a function expression");

Once you define a function using this syntax, the function name [logMessage in this case] becomes obsolete and the function can be called only using the assigned variable name [log]. So in a nutshell you need to follow some rules to define function expression –

1. Function expression cannot start with word ‘function’.
2. Function expression can be a named function or an anonymous function.

So you might be thinking, JavaScript runtime must be doing function hoisting in this case as well and you can call the function first and define it later? Well, your assumption is partially correct. JavaScript engine does perform function hoisting in this case as well, however since the function was assigned to a variable value, it hoist the variable instead of complete function definition. So JavaScript runtime re-implements the code as shown in below code snippet

var log;

log = function logMessage(message){

console.log(message);

}

log("This is a function expression");

Note again, it doesn’t hoist function definition completely. It just hoists the assigned variable and initializes it where function was initially defined. So in this case, you cannot call the function unless it’s explicitly defined earlier. Doing so will result into a runtime exception ‘undefined is not a function’ since we are trying to call a function which is not defined.

var log;

log("JavaScript is my favorite language");

log = function logMessage(message){

console.log(message);

}